**Cloud Custodian Serverless Deployment**

Workflow Diagram



The Solution

**Summary** leverages the Open Source tool Cloud Custodian to automate the operation tasks for the EC2 fleet and S3 buckets. When a user runs Custodian, Custodian will run the specified policy against the account and region specified by the user. Custodian will iterate over all resources defined in the policy. You can find more details in http://www.capitalone.io/cloud-custodian/docs/overview/index.html# Custodian supports enforcing the policy in CLI, Lambda or Config mode and it can be trigger manually, periodically or by CloudWatch Events. In this POC, we opt for the serverless approach where the Custodian policy will be deployed by the CodePipeline and enforced in Lambda periodically. (set at 1 minute interval and should be adjust accordingly for your envionrment)

**To Deploy**the Custodian Pipeline including the CodeBuild project and the required IAM Services roles and S3 buckets are created atomically by the “CustodianPolicyDeploymentPipeline.yml” CloudFormation template. To deploy the solution, you will need to create a source artifact S3 bucket (where CodePipeline poll for change) and a custodian output log S3 bucket (where Custodian sends the execution output – needs to match the name specified in the Custodian Policy under mode -> execution-options)

**Source Control**enable S3 Versioning to provide version control and implement Bucket/IAM policy to secure both the source and output buckets. The S3 buckets can be in a different AWS account than the account where the Pipeline resides. (need to grant get object permission to the CodePipeline Service role) To modify the policies, always download the latest version from the source S3 bucket to ensure consistency.

**To Use** 5 steps involved to deploy the policies

1. download the custodian.zip file from the Source S3 bucket
2. create/edit the Custodian policy files \*\*you need to make sure both the “output\_dir” and “role” under the “mode” session matches your particular deployment
3. edit the buildspec.yml file. Define the Custodian commands you want to run under the “build” session of the file. (i.e. custodian run -s output s3globalgrant-lambda-policy.yml --region us-east-2)
4. zip the buildspec and all the policies file and rename it to “custodian.zip”
5. upload the zip file to the Source S3 bucket.

Use Cases in the POC

**Tagging Compliance** 4 policies defined in the “cat-ec2tagcompliance-stop-policy.yml” file.

ec2-tag-compliance-mark

check the non-ASG running instances’ EC2 tag and mark it for stoppage in 5 days (can be adjusted) if found non-compliant.

ec2-tag-compliance-unmark

remove the “stop” marking if the user fixes the tagging issue and becomes compliant

ec2-tag-compliance-report (optional)

create a custom Config rule to show all the non-compliances instances

ec2-tag-compliance-stop (optional)

stop all the instances previously marked for stoppage by today’s date every minute. (can be adjust)

**S3 Permission Compliance** 1 policy defined in the “s3-global-access-lambda.yml” file. It will scan all the S3 buckets with permission set to allow either “All user” or “All authenticated Users” for any access and remove them atomically.

**EC2 Scheduling** 1 policy defined in the sample-ec2offhour-optin-lambda-policy.yml” file.It will start/stop EC2 instances base on the schedule defined in the policy (8am-2pm CST M-F). The solution is opt-in only, meaning it will take effect only on the EC2 instances with the the “maid\_offhours” tag and value empty or set to “on”. Individual EC2 instance can also overwrite the policy default schedule using the “maid\_offhours” tag and customized the hours (i.e. maid\_offhours =off=(M-F,18);on=(M-F,8);tz=cdt) You can also change the policy to deploy in opt-out model or customize the tag to a different name.

Note when custodian runs, if it’s 6:00pm or 6:59 pm CDT time, it will shut down the VM you tagged this way. The key is the hour integer matching 18. If custodian runs at 5:59pm or 7:00pm, it won’t shut down the VM. Same idea for starting.

Cross Region/Cross Account

**Cross Region** you specify which region or regions you want to run the policies (Lambda) in the buildspec.yml file (i.e. custodian run -s output s3globalgrant-lambda-policy.yml --region us-east-2 --region us-east-1) Note that separate Lambda functions will be deployed per region, meaning 4 lambda functions will be deploy if you rollout 2 Custodian policies in 2 regions.

**Cross Account** Custodian supports cross account deployment by ways of STS assume role. (not covered in the POC) You specify the role under the Custodian policy file under mode -> execution –options-> assume\_role and supply the role in ARN format.

* + Config: May run in a different region but not cross-account
  + Event: Only run in the same region and account
  + Periodic (Lambda): May run a different region and different account

Monitoring

**CloudWatch** by default Cloud Custodian generates CloudWatch metrics on each policy for the number of resources that matched the set of filters, the time to retrieve and filter the resources, and the time to execute actions. CloudWatch metrics is turn-on by default when running in Lambda mode. You can optionally configure Custodian to send a copy of the logs in realtime to CloudWatch log by specifying the log group in the policy.

**S3 Logs & Records** Custodian will output its logs and structured resource records in JSON format to S3, along with its log files for archival purposes. You can specify the S3 bucket location in the Custodian policy file under mode -> execution –options-> output\_dir (i.e. s3://tac-cc-outputlogs) Enable version control or the output will be overwritten per day.